WeCAR
Lane_detection_WeCAR

WeGo Korea

목차

01

1. Lane detection_WeCAR

VESC

- 차선인식_pkg

lane_detection_wecar

- include
- launch
- scripts
- src
- CMakeLists.txt
- package.xml

ROS

WeGo

Detection_in_pkg

- 차선인식_CMakeList

CMakeLists.txt

D: > WeCAR - 군산대학교 > pkg > lane_detection_wecar > CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(lane_detection)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)
```

Detection_in_pkg

- 차선인식_package.xml_(1)

```xml
<?xml version="1.0"?>
<package format="2">
  <name>lane_detection</name>
  <version>0.0.0</version>
  <description>The lane_detection package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="nvidia@todo.todo">nvidia</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>
```

ROS

WeGo

Detection_in_pkg

- 차선인식_package.xml_(2)

```
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>rospy</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>rospy</exec_depend>
  <exec_depend>std_msgs</exec_depend>


  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->

  </export>
</package>
```

ROS

WeGo

Detection

- image_data.py

```python
import cv2
import math
import numpy as np

# ignore this function
def nothing(x):
    pass
#====================

# Crop the image method
def region_of_interest(img, roi):
    vertices = np.array([roi], np.int32)
    mask = np.zeros_like(img)

    #channel_count = img.shape[2]
    #match_mask_color = (255,) * channel_count

    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)

    masked_image = cv2.bitwise_and(img, mask)
    return masked_image
```

ROS

WeGo

Detection

- image_data.py

(전체 코드가 아닌)

참고용_FYI

```python
def find_edges(img, h_min = 0, h_max = 0, s_min = 0, s_max = 0, v_min = 0, v_max = 0):
    height = img.shape[0]
    width = img.shape[1]

    # Setting ROI value
    # When you want to Change Detect Reach,
    # Change the below value
    roi = [
        (0, height),
        (0, height / 2),
        (width, height / 2),
        (width, height)
    ]

    # Blur
    img = cv2.GaussianBlur(img, (5, 5), 0)

    # Change image channel to HSV
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    low_val = (h_min, s_min, v_min)
    high_val = (h_max, s_max, v_max)

    # Threshold
    mask = cv2.inRange(hsv, low_val, high_val)

    # Crop the image
    mask = region_of_interest(mask, roi)

    # find edges
    edges = cv2.Canny(mask, 75, 150)

    return edges
```

ROS

WeGo

Detection

- image_data.py

(전체 코드가 아닌)

참고용_FYI

```python
def find_lines(image, lines):
    width = image.shape[1]

    # init list
    lane_lines = []

    left_fit = []
    right_fit = []

    # When you want to
    boundary = 2 / float(3)
    left_region_boundary = width * (1 - float(boundary))
    right_region_boundary = width * float(boundary)

    for line in lines:
        for x1, y1, x2, y2 in line:
            if x1 == x2:
                continue
            # find a linear function
            fit = np.polyfit((x1, x2), (y1, y2), 1)
            slope = fit[0]
            intercept = fit[1]

            if slope < 0:
                if x1 < left_region_boundary and x2 < left_region_boundary:
                    left_fit.append((slope, intercept))
            else:
                if x1 > right_region_boundary and x2 > right_region_boundary:
                    right_fit.append((slope, intercept))

    if len(left_fit) > 0:
        left_fir_average = np.average(left_fit, axis=0)
        lane_lines.append(make_points(image, left_fir_average))
    if len(right_fit) > 0:
        right_fit_average = np.average(right_fit, axis=0)
        lane_lines.append(make_points(image, right_fit_average))
    return lane_lines
```

ROS

WeGo

Detection_in_pkg

- Wecar_lane_detection_ROS.py (1)

```python
#!/usr/bin/env python
from __future__ import print_function

import roslib
import sys
import rospy
import cv2
from std_msgs.msg import String
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from std_msgs.msg import Float64

from image_data import *

class image_converter:

  def __init__(self):
    # self.image_pub = rospy.Publisher("image_topic_2",Image)
    self.rate = rospy.Rate(20)
    self.bridge = CvBridge()
    self.timer_to_sending_data = 0


    self.speed = rospy.Publisher('/commands/motor/speed', Float64, queue_size=1)
    self.position = rospy.Publisher('/commands/servo/position', Float64, queue_size=1)

    self.speed_value = 1200
    self.position_value = 0.5
   # self.speed.publish(0.0)
    self.position.publish(self.position_value)
    self.image_sub = rospy.Subscriber("/usb_cam/image_raw",Image,self.callback)
    rospy.on_shutdown(self.shutdown)
```

ROS

WeGo

Detection_in_pkg

- Wecar_lane_detection_ROS.py (2)

```python
def callback(self,data):
    self.rate.sleep()
    try:
        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
    except CvBridgeError as e:
        print(e)

# cv2.imshow("Image window", cv_image)
# cv2.waitKey(1)

#self.speed.publish(self.speed_value)
# self.position.publish(self.position_value)
if self.timer_to_sending_data %5 == 0:

    angle = line_detection(cv_image)
    print(angle)

    if angle > 0:
        self.speed_value = 1000
    self.position_value = 0.8
    self.position.publish(self.position_value)
    self.speed.publish(self.speed_value)
    elif angle < 0:
        self.speed_value = 1000
    self.position_value = 0.0
    self.position.publish(self.position_value)
    self.speed.publish(self.speed_value)
    else:
        self.position_value = 0.4
    self.speed_value = 1200
    self.speed.publish(self.speed_value)
    self.position.publish(self.position_value)
    self.timer_to_sending_data = 0
```

ROS

WeGo

Detection_in_pkg

- Wecar_lane_detection_ROS.py (3)

```
        self.timer_to_sending_data += 1

    def shutdown(self):
        self.speed.publish(0)
        self.position.publish(0.4)
        self.rate.sleep()

def main(args):


    rospy.init_node('image_converter', anonymous=True)


    ic = image_converter()

    #except Exception:
        #ic.shutdown()
        # print("Shutting down")

    rospy.spin()

if __name__ == '__main__':
    main(sys.argv)
```

ROS

WeGo

02

2. Lane detection_Application

Lane_detection_Application

ROS Package_Problem Camera응용

* openCV를 이용하여 차선 인식